Сергей Дронский

МОДУЛЬ EHS6 И МАКЕТНАЯ ПЛАТА CINTERION CONCEPT BOARD В РУКАХ СПЕЦИАЛИСТА

В статье описан опыт начала работы с макетной платой Cinterion Concept Board Gemalto, на которой установлен однокорпусной модуль (System In a Package, SIP) Cinterion EHS6.

Общее описание однокорпусной системы EHS6

EHS6 – это модуль пятого поколения для построения интерфейсов типа M2M (Mashine to Mashine). В его основе лежит архитектура ARM11, на базе которой работает исполнительный модуль Java ME 3.2, оптимизированный для ограниченных в ресурсе M2M-приложений. Наличие большого количества готовых программных модулей позволяет сократить время разработки и уменьшить стоимость готовых изделий. Raspberry Pi также использует процессор на базе архитектуры ARM11.

По сути, EHS6 представляет собой сотовый модем с Java-машиной. Управлять системой можно как снаружи, AT-командами (например от внешнего процессора), так и Java-программой (изнутри). Модуль содержит сотовый интерфейс (пятидиапазонный 3G HSPA, четырехдиапазонный GPRS\EDGE Class12). На макетной плате имеется встроенная антенна сотовой связи и держатель SIM-карты стандартного размера с функцией определения ее наличия. Скорость передачи данных «вниз» достигает 7,2 Mbps, а «вверх» – 5,7 Mbps. Модуль поддерживает многопоточное и многопротокольное выполнение Java midlet (Multi MIDlet Java execution). Программе доступны 6 Мбайт RAM и 10 Мбайт Flash-памяти.

Технические характеристики EHS6 приведены в таблице 1. Модуль выполнен в корпусе LGA, мощность потребления и тепловыделение оптимизированы для промышленного применения в тяжелых условиях.

Таблица 1. Технические характеристика модуля EHS6

Частотный диапазон, *МГц*	800/850/900/1900/2100
Интерфейсы	2xUART, USB 2.0, SPI, I²C, 16xGPIO, SIM-интерфейс 1,8/3,3 В, цифровой звук
Специальные функции	Встроенный TCP/IP-стек Java ME Embedded 3.2 Мониторинг сетей без SIM-карты Jamming Detection (обнаружение помех) FOTA (обновление прошивки модема и ПО пользователя «по воздуху»)
Габариты, *мм*	27,6x25,4x2,2
Напряжение питания, *В*	3,3–4,5
Диапазон рабочих температур	–30…+85 °C

Интерфейсы и вводы/выводы

16 выводов общего назначения (GPIO), доступных из Java-мидлета, могут быть использованы для нескольких интерфейсов (сигналы DSR, DTR, DCD интерфейса ASC0; RXD, TXD, RTS, CTS интерфейса ASC1; SPI; Fast-Shutdown; Network-Status-Indication; PWM; Pulse-Counter lines).

В таблице 2 приведены варианты использования GPIO и разных интерфейсов. Если вывод применяется для работы в составе какого-либо интерфейса, то как вывод общего назначения его уже нельзя использовать. Например, если в приложении используется SPI, то выводы 3, 16, 17, 19 станут недоступными в качестве портов ввода/вывода общего назначения.

Цифровой аудиоинтерфейс, который можно использовать для передачи аудиоданных по каналу связи, требует использования внешнего соответствующего кодера/декодера.

Температурный контроль

В модуле присутствуют встроенные механизмы управления температурой (Advanced temperature management). При выходе температуры за допустимые пределы (–40…+90 °C)

Таблица 2. Варианты использования GPIO и различных интерфейсов

Table 6: GPIO lines and possible alternative assignment

GPIO	Fast Shutdown	Status LED	PWM	Pulse Counter	ASC0	ASC1	SPI	HSIC	PCM
GPIO1					DTR0				
GPIO2					DCD0				
GPIO3					DSR0		SPI_CLK		
GPIO4	FST_SHDN								
GPIO5		Status LED							
GPIO6			PWM2						
GPIO7			PWM1						
GPIO8				COUNTER					
GPIO11									
GPIO12									
GPIO13									
GPIO14									
GPIO15									
GPIO16						RXD1	MOSI	AP_WAKEUP	
GPIO17						TXD1	MISO	HOST_ACTIVE	
GPIO18						RTS1		CP_WAKEUP	
GPIO19						CTS1	SPI_CS	SUSPEND	
GPIO20									TXDDAI
GPIO21									RXDDAI
GPIO22									TFSDAI
GPIO23									SCLK
GPIO24					RING0				

модуль выключится с выдачей соответствующего сообщения. Температура модуля контролируется NTC-резистором, установленным на плате, и может не соответствовать внешней температуре. Если выдача предупреждающих сообщений разрешена, то при возникновении угрожающего состояния будет выдано такое сообщение. Угрожающее состояние – подход близко к перегреву или переохлаждению. В этом состоянии модуль продолжит работу. Проектировщик системы может предусмотреть обработку таких сообщений и реакцию на них: например, включить подогрев или охлаждающий вентилятор (или холодильник на элементах Пельтье) для возврата температуры системы в норму. Если же температура модуля выйдет за пределы безопасной эксплуатации, то модуль выдаст тревожное сообщение и выключится. Запретить выдачу тревожного сообщения о перегреве или переохлаждении нельзя.

Определение подавления радиосигнала

Рассмотрим механизм определения подавления радиосигнала (RLS monitoring\jamming detection). Если модуль используется в составе охранной системы и сигнал сотовой связи подавляется внешними устройствами, то модуль определит факт такого подавления. Выполняемая программа сможет отреагировать на факт недоступности канала связи и изменить ход выполнения в соответствии с заложенной программистом логикой.

Обновление программного обеспечения

Механизм загрузки обновленных программ через сотовый интерфейс (FOTA) можно настраивать под собственные потребности. Он бесплатен, так как представляет собой Java-мидлет с открытым кодом. Возможности FOTA в EHS6 очень широки: можно обновлять не только ПО модуля но и Java-мидлеты пользователя; ПО можно загружать с обычного HTTP/FTP-сервера; команда на обновление может быть как локальной, так и удаленной (через IP-соединение или SMS). Наличие такого механизма предоставляет широчайшие возможности для разработчика по удаленному управлению модулем и системой на базе такого модуля. Можно корректировать ошибки и загружать исправленные версии, можно добавлять новые возможности в программу и т. д. И, что особенно приятно, автоматизированно загружать исправленное ПО без вмешательства конечного пользователя.

Макетная плата Cinterion Concept board

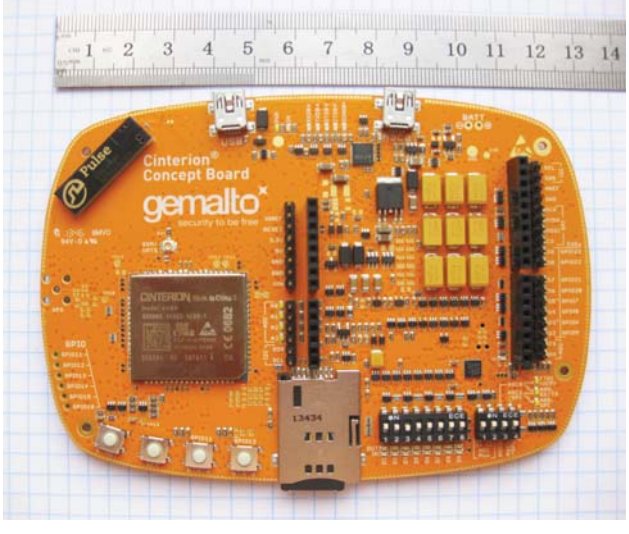

Рис. 1. Внешний вид макетной платы Cinterion Concept Board Gemalto

На макетной плате Cinterion Concept Board (рис. 1) смонтирован модуль EHS6 и вспомогательная электроника. Структурная схема показана на рис. 2.

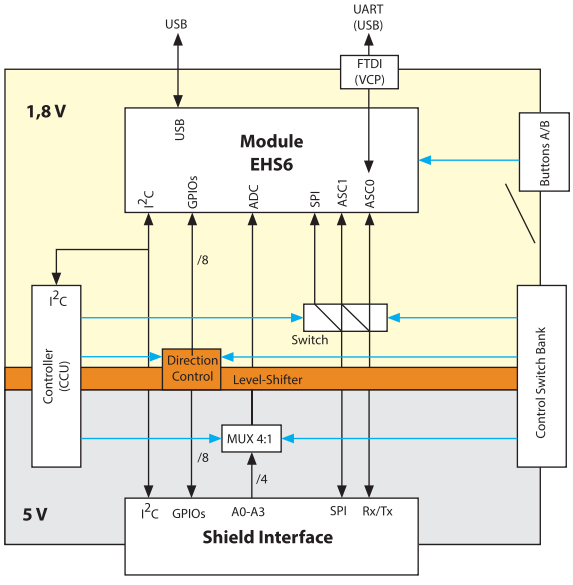

Рис. 2. Структурная схема макетной платы Cinterion Concept Board Gemalto

Разработчики макетной платы предусмотрели физическую совместимость с интерфейсом популярного семейства «Ардуино»: физические размеры разъема, электрические параметры и набор сигналов позволяют прямое подключение платы «Ардуино» и/или разнообразных датчиков из бесчисленного множества предназначенных для нее устройств.

На плате установлен вспомогательный микропроцессор типа STM8L151GX. Разработчики назвали его CCU – configuration and control unit, что можно перевести как «устройство конфигурирования и контроля». CCU позволяет программно управлять направлением каналов ввода/вывода, выбирать один из четырех каналов аналого-цифрового преобразователя и выбирать либо ASC1 либо канал SPI.

Выбор делается либо переключателем на плате, либо программно. Может возникнуть ситуация, когда переключатель и CCU дают разные команды. Управление от CCU всегда имеет более высокий приоритет над микропереключателями. Состояние вывода (вход или выход) отображается светодиодом на плате всегда правильно, независимо от положения соответствующего переключателя.

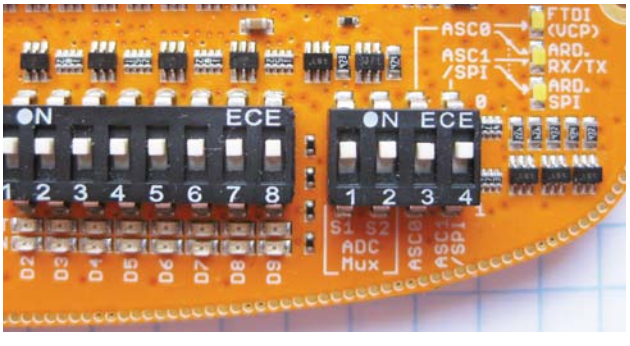

Рис. 3.

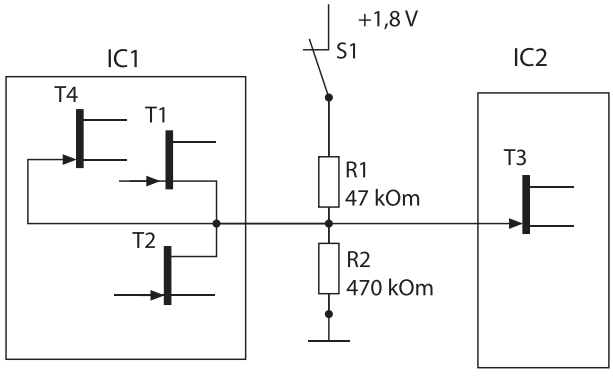

Рис. 4. Упрощенная схема коммутации

Упрощенная схема коммутации представлена на рис. 4. Рассмотрим, как она работает.

Режим ручного управления. Выход микросхемы IC1 находится в третьем состоянии и не влияет на потенциал в точке соединения резисторов R1 и R2. Потенциал этой точки определяется состоянием контактов микропереключателя S1. По умолчанию (в режиме заводской поставки) контакты микропереключателя S1 замкнуты, что дает высокий потенциал на входе микросхемы IC2, и схема преобразования направления и уровней находится в режиме «выход», обеспечивая на GPIO сигналы в диапазоне от 0 до +5 В.

При разомкнутом переключателе схема согласования уровней переходит в режим входа, обеспечивая трансляцию уровней 0…+5 В в уровни 0–1,8 В.

Режим программного управления. Для перехода в режим программного управления выход микросхемы IC1 надо переключить из режима третьего состояния в режим активного выхода. Поскольку выходной импеданс микросхемы пренебрежимо мал по сравнению с импедансом резисторного делителя (разница примерно на три порядка), то уровень сигнала на выходе системы управления будет целиком определяться состоянием выхода микросхемы IC1. А состояние контактов ключа S1 влиять на выходной сигнал не будет.

Транзистор T4 показан, чтобы напомнить о возможности перевести вывод микросхемы IC1 в режим ввода и программно опросить состояние микропереключателя S1.

CCU подключен к модулю EHS6 по каналу I^2C. Адрес CCU – 0x69. Линии интерфейса I^2C (кроме CCU) также выведены на разъем и позволяют подключение внешних I^2C-контроллеров. Линии I^2C также выведены на внешний разъем через двунаправленные преобразователи уровней 1,8–5 В на основе полевых транзисторов.

Модуль CCU может управлять направлением передачи информации в портах GPIO0, GPIO7 выбором одного из четырех каналов АЦП, подключением порта ASC0 к разъему «Ардуино» или к преобразователю FT232 и подключением порта ASC1 либо к выводам сериал порта на разъеме «Ардуино», либо к выводам SPI.

На макетной плате имеется коммутатор аналогового сигнала 0–5 В в один канал 0–1,2 В. Выбор канала можно делать либо блоком микропереключателей, либо программно. ADC в EHS6 может преобразовывать напряжение максимум в 1,2 В. Для расширения входного диапазона до 5 В применен преобразователь на операционном усилителе. Для корректной обработки результатов следует использовать коэффициент 4,17. Так, например, если в Java-программе считан результат 1 В, то его следует умножить на 4,17 и этот результат как раз и будет истинным входным напряжением.

Сериал-порт ASC0 модуля EHS6 может быть подключен системой управления (CCU + микропереключатели) либо напрямую к выводам разъема «Ардуино», либо к микросхеме трансляции сериал-порта в USB FTDI FT232RQ. Интерфейс ASC0 содержит девять сигналов для управления потоком данных. Можно использовать подключение через FT232 по USB с драйвером виртуального сериал-порта. Питание модуля также возможно через второй мини-USB-разъем.

Выводы GPIO3, GPIO16-19 модуля EHS6 могут быть использованы либо как сериал-порт ASC1, либо как SPI. В обоих случаях автоматически обеспечивается нужное направление преобразования сигналов 1,8–5 В для согласования электрических параметров интерфейса. Для включения интерфейса ASC1 или SPI нужно в программе выполнить две операции: дать соответствующую команду CCU и создать SPI-соединение и потоки данных. Интерфейс ASC1 содержит четыре сигнала: RX, TX, RTS,CTS.

На макетной плате предусмотрены выводы для подключения внешнего литиевого аккумулятора. Зарядка батареи происходит током порядка 250 мА. При выборе аккумулятора следует иметь в виду, что при работе модуля на передачу он потребляет значительный ток (порядка 2 А), и нельзя допустить просадку напряжения ниже 3,3 В, так как при этом модуль отключится. Изготовитель рекомендует емкость аккумулятора не ниже 800 мА•ч.

В модуле EHS6 имеются часы реального времени, но для их работы нужно, чтобы к макетной плате был постоянно подключен аккумулятор.

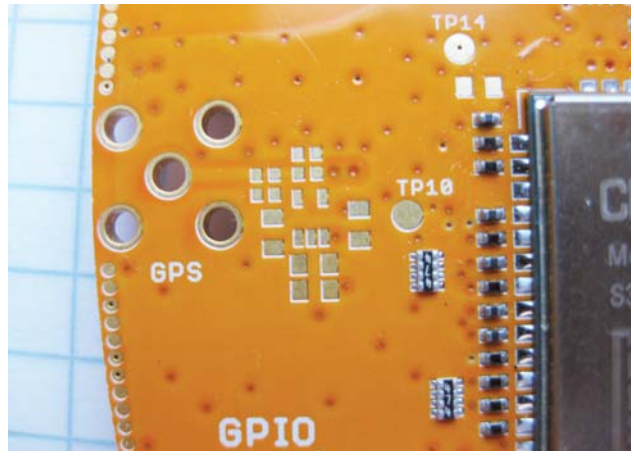

Рис. 5. Места под установку вспомогательных элементов

Ни EHS6, ни макетная плата не поддерживают прерываний. Единственный способ реакции на изменение входных сигналов – опрос.

На макетной плате также имеются места под установку разъема и вспомогательных радиоэлементов, помеченных «GPS» (рис. 5). В производственной гамме Gemalto имеются модули с поддержкой ГЛОНАСС и даже ЭРА-ГЛОНАСС (например модуль EHS5). Этот модуль полностью совместим по выводам с описываемым EHS6.

Также на плате имеется U-Fl-разъем для подключения внешней сотовой антенны. По спецификации модуль EHS6 полностью защищен от несогласованной антенны и способен выдержать совсем несогласованную антенну без повреждений. Учитывая, что по спецификации модуль способен отдавать в нагрузку до 2 Вт и если нагрузка предельно не согласована, то вся эта мощность будет рассеяна внутри модуля. Модуль оснащен встроенными средствами контроля температуры и должен будет при перегреве отключиться. Однако всегда следует помнить один из законов Мэрфи: «транзистор, защищенный предохранителем, вылетит первым, защитив предохранитель!» – и быть аккуратным с внешней антенной. Ну и лишнее СВЧ-излучение организму тоже неполезно.

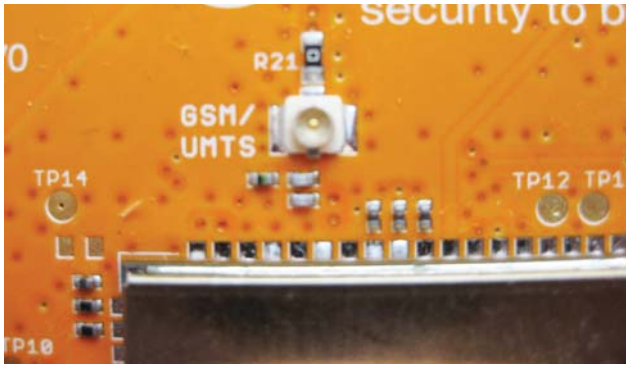

Рис. 6.

На макетной плате отсутствуют источники питания для поддержки часов реального времени, чтобы получить такую поддержку – надо припаять литиевую батарею к выводам BATT.

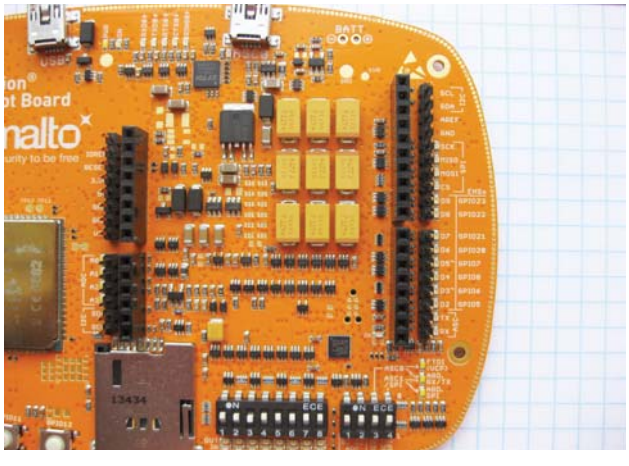

Рис. 7.

На макетной плате установлены четыре кнопки (рис. 8).

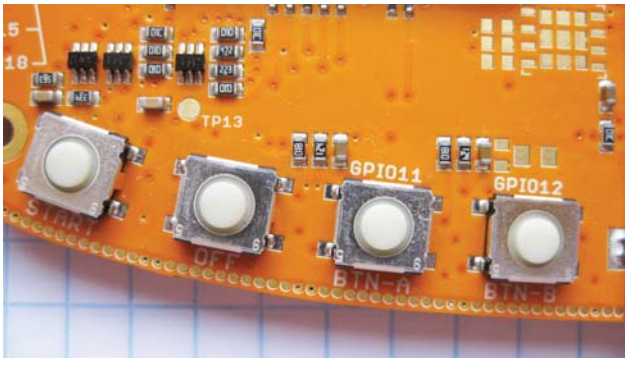

Рис. 8.

Кнопка Start служит для подачи сигнала Ignition для EHS6, кнопка OFF выключает модуль. Оставшиеся две кнопки (BTN-A и BTN-B) подключены к выводам GPIO11 и GPIO12 и могут произвольным образом использоваться в программе пользователя.

Установка программного обеспечения

Для начала следует зарегистрироваться на http://m2m.gemalto.com/ и указать IMEI-номер системы. Этот номер находится на модуле EHS6 в нижней строке (рис. 9), в данном случае это 358884 05 007431 4.

Рис. 9. IMEI-номер системы

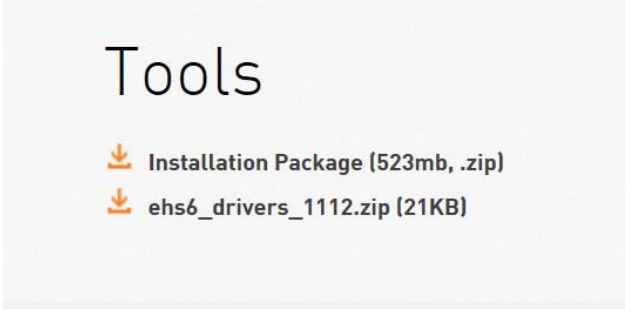

Рис. 10. Файлы, необходимые для установки ПО

Через некоторое время (в моем случае ожидание заняло около двух часов) приходит письмо с логином и паролем. С их помощью осуществляем вход в систему и заходим в Cinterion® Concept Board Support. Нам нужны файлы, названия которых показаны на рис. 10.

Распаковываем Installation Package и производим установку путем запуска Setup. Я поставил систему на виртуальную машину под управлением Windows 7 (32 бит), но, по информации специалистов Gemalto, можно использовать и 64-бит систему.

В процессе установки надо будет подключить плату и нажать кнопку «Старт». В моем случае драйвера платы установились автоматически. Но если от вас потребуется указать драйвера, они находятся в файле eh6_drivers_1112.

Установщик не размещает ссылку на eclipse.exe нигде, так что это нужно сделаь вручную. Программа лежит по такому пути: C:\Users\Public\Eclipse\eclipse\eclipse.exe

В системе появятся новые устройства (рис. 11).

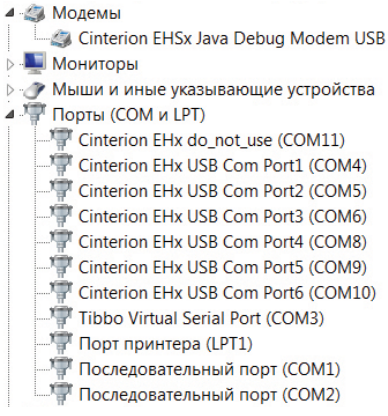

Рис. 11. Устройства, появившиеся после установки драйверов платы

Для общения с устройством по сериал-интерфейсу можно воспользоваться какой-либо доступной программой, например Putty (рис. 12).

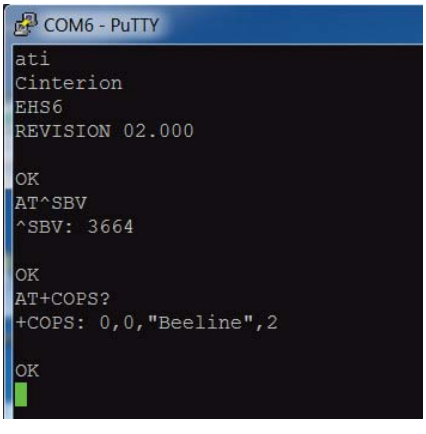

Рис. 12. Окно «общения» с устройством

Первая команда – запрос идентификации. Вторая – запрос напряжения питания, в данном случае это 3,664 В. Третья – запрос регистрации в сотовой сети. В нашем случае модуль зарегистрирован в сети «Билайн».

После успешной установки Eclipse можно импортировать в Workspace демонстрационные программы и исследовать возможности Java-мидлетов. Демонстрационных программ вполне достаточно, чтобы показать работу системы и взаимодействие приложения с аппаратурой на плате. Есть традиционный Hello World и не менее традиционное моргание светодиодом на плате.

Импорт готовых проектов делается так, как показано на рис. 13.

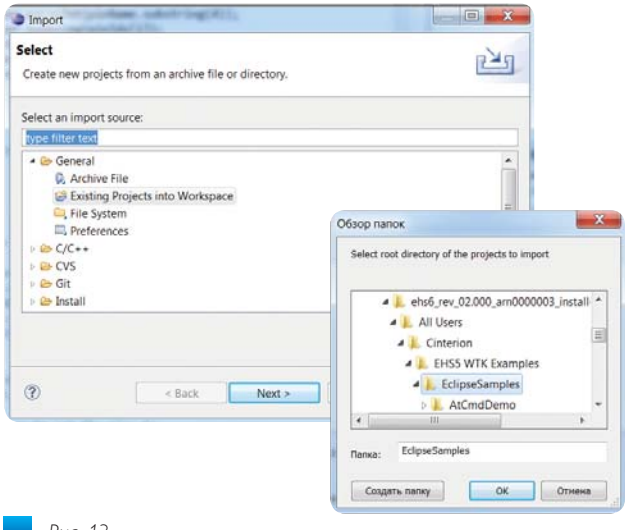

Рис. 13.

В результате получаем вот такое рабочее пространство, показанное на рис. 14.

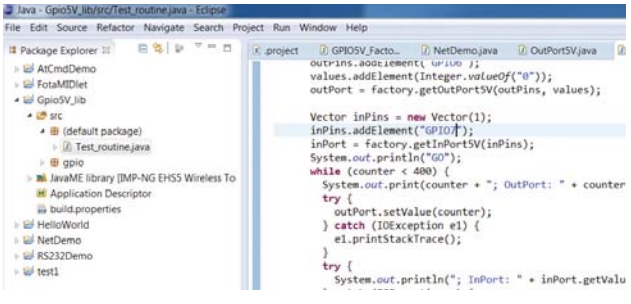

Рис. 14.

Можно запустить традиционный Hello World и посмотреть на вывод в консоль (рис. 15).

Можно запустить программу мигания светодиодом Test_routine из проекта GPIO5v_lib. С ней будет не все так гладко. В этой программе не сделан перевод вывода GPIO7 в режим ввода. Поэтому вывод будет выглядеть вот так, как показано на рис. 16. А на самой плате Cinterion будет мигать только один светодиод.

```
Test_routine [Emulated JavaME MIDlet] cmd (13.09.14 13:0
Installing suite from: http://192.168.244.2
GO
0; OutPort: 0; InPort: 0; ADC: 0.0 mV
1; OutPort: 1; InPort: 0; ADC: 0.0 mV
2; OutPort: 0; InPort: 0; ADC: 0.0 mV
3; OutPort: 1; InPort: 0; ADC: 0.0 mV
4; OutPort: 0; InPort: 0; ADC: 0.0 mV
5; OutPort: 1; InPort: 0; ADC: 0.0 mV
6; OutPort: 0; InPort: 0; ADC: 0.0 mV
7; OutPort: 1; InPort: 0; ADC: 0.0 mV
8; OutPort: 0; InPort: 0; ADC: 0.0 mV
9; OutPort: 1; InPort: 0; ADC: 0.0 mV
10; OutPort: 0; InPort: 0; ADC: 0.0 mV
11; OutPort: 1; InPort: 0; ADC: 0.0 mV
12; OutPort: 0; InPort: 0; ADC: 0.0 mV
13; OutPort: 1; InPort: 0; ADC: 0.0 mV
14; OutPort: 0; InPort: 0; ADC: 0.0 mV
```

Рис. 16.

Как видим, сигнал InPort всегда «0», а должен быть равным сигналу OutPort. Исправить ситуацию можно двумя путями: либо использовать микропереключатель, либо сделать это программным путем. Для ручного переключения GPIO7 в режим ввода надо перевести микропереключатель D5 в нижнее положение. Вместо желтого светодиода включится рыжий.

Практический пример

Подробное описание протокола обмена с CCU содержится в документе concept_board_hd_v01a.pdf. Его можно загрузить с сайта http://m2m.gemalto.com/, который становится доступным после регистрации. Здесь я дам только практическую «выжимку» для конкретного случая.

Адрес CCU на шине I2C 0x69 или 1101001 в двоичном виде. Команда запись ("1") или чтение ("0") добавляется в конец посылки и становится восьмым битом. Таким образом, получаем либо 0xD2 для записи, либо 0xD3 для чтения.

Следующий байт – адрес внутреннего регистра CCU.

Нас интересует GPIO7, его адрес 0x13.

И последний байт посылки – команда включить канал GPIO на ввод 0x00 (на вывод – 0x01 соответственно).

Итак, полная команда для перевода GPIO7 в режим чтения будет выглядеть так: aD21300.

Префикс "a" в начале строки – Message ID, может быть только одной буквой.

Допишем в текст программы следующие строки:

```
I2cBusConnection cc = (I2cBusConnection) Connector.
    open("i2c:0;baudrate=400");
//int baudrate = cc.getBaudRate();
InputStream inStream  = cc.openInputStream();
OutputStream outStream = cc.openOutputStream();

String data = "<aD21300>";

outStream.write(data.getBytes(), 0, data.length());
outStream.flush();

byte[] inBuf = new byte[12];
if (inStream.available()>0)
inStream.read(inBuf);

inStream.close();
outStream.close();
cc.close();
```

Теперь в процессе выполнения программы канал GPIO7 будет автоматически переключен на ввод и программа целиком заработает так, как задумано без необходимости манипулировать вручную микропереключателями (рис. 17).

```
Problems  @ Javadoc  Declaration  Console
Test_routine [Emulated JavaME MIDlet] cmd (13.09.14 12:47)
9; OutPort: 1; InPort: 1; ADC: 4753.8 mV
10; OutPort: 0; InPort: 0; ADC: 4753.8 mV
11; OutPort: 1; InPort: 1; ADC: 4749.63 mV
12; OutPort: 0; InPort: 0; ADC: 4757.97 mV
13; OutPort: 1; InPort: 1; ADC: 4753.8 mV
14; OutPort: 0; InPort: 0; ADC: 4757.97 mV
15; OutPort: 1; InPort: 1; ADC: 4753.8 mV
16; OutPort: 0; InPort: 0; ADC: 4753.8 mV
17; OutPort: 1; InPort: 1; ADC: 4753.8 mV
18; OutPort: 0; InPort: 0; ADC: 4757.97 mV
19; OutPort: 1; InPort: 1; ADC: 4749.63 mV
20; OutPort: 0; InPort: 0; ADC: 4757.97 mV
21; OutPort: 1; InPort: 1; ADC: 4749.63 mV
22; OutPort: 0; InPort: 0; ADC: 4753.8 mV
23; OutPort: 1; InPort: 1; ADC: 4753.8 mV
24; OutPort: 0; InPort: 0; ADC: 4749.63 mV
25; OutPort: 1; InPort: 1; ADC: 4749.63 mV
26; OutPort: 0; InPort: 0; ADC: 4749.63 mV
```

Рис. 17.

Соответственно получим синхронное мигание светодиодов (рис. 18).

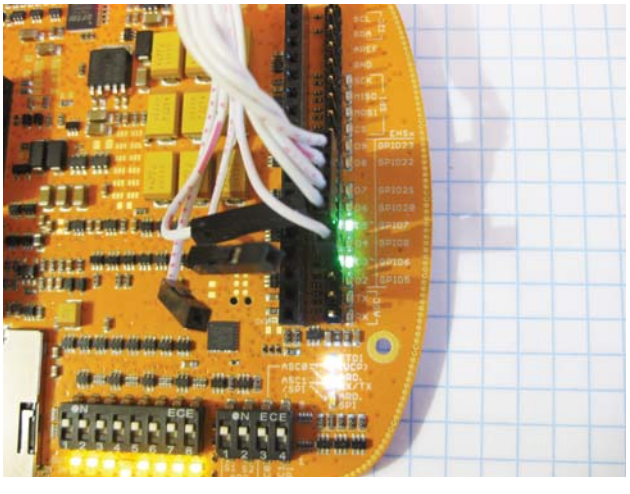

Рис. 18.

Обратите внимание на положение микропереключателя D5, управляющего направлением порта GPIO7. Он стоит в положении «вывод», но включен рыжий нижний светодиод: программно от CCU включен режим «ввод» и команда от CCU имеет более высокий приоритет.

Также я добавил в программу чтение показаний ADC:

ADC adc = new ADC(0,0);

И в основной цикл:
System.out.println("; ADC: " + adc. getValue() *4,17 + " mV");

Не забываем умножить на 4,17 – учитывая поправочный коэффициент преобразователя напряжения на входе. Полученные 4,75 В – это как раз напряжение питания минус падение напряжения на развязывающих элементах.
Добавим считывание нажатия кнопок на макетной плате и изменим программу так, чтобы она останавливалась по нажатию любой из кнопок.

Добавим в текст программы строки:
Vector pins1 = new Vector(2);

```
pins1.addElement("GPIO11");
pins1.addElement("GPIO12");

InPort inport1 = new InPort(pins1);
```

Изменим условие продолжения цикла:
while (inport1.getValue()==0)
Теперь программа крутится бесконечно, пока не будет нажата любая из кнопок на макетной плате.
И в качестве «вишенки на торте» добавим посылку SMS после нажатия любой из кнопок. Такая программа может рассматриваться как proof of concept работы системы, реагирующей на срабатывание внешнего датчика и посылающего SMS, информирующие пользователя о возникновении некоего условия. Модуль EHS6 правильно работает с AT-командами посылки текстовых SMS, что позволило не заниматься генерацией PDU-сообщения и решить задачу простым путем. В функции посылки SMS предварительно проверяется статус модуля – зарегистрирован ли он в сотовой сети.
Текст программы, в котором красным выделены внесенные изменения:

```
/*
 *
 * Copyright (C) Gemalto M2M GmbH 2013. All Rights reserved.
 *
 * Gemalto M2M GmbH ("Gemalto") grants Licensee a non-exclusive,
 * non-transferable, limited license to transmit, reproduce, disseminate, utilize
 * and/or edit the source code of this Software (IMlet, LIBlet, batch files,
 * project files) for the sole purpose of designing, developing and testing
 * Licensee's applications only in connection with a Gemalto Wireless Module.
 *
 * THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
 * EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE.
 *
 * GEMALTO, ITS LEGAL REPRESENTATIVES AND VICARIOUS AGENTS SHALL - IRRESPECTIVE
 * OF THE LEGAL GROUND - ONLY BE LIABLE FOR DAMAGES IF THE DAMAGE WAS CAUSED
 * THROUGH CULPABLE BREACH OF A MAJOR CONTRACTUAL OBLIGATION (CARDINAL DUTY),
 * I.E. A DUTY THE FULFILMENT OF WHICH ALLOWS THE PROPER EXECUTION OF THE
 * RESPECTIVE AGREEMENT IN THE FIRST PLACE OR THE BREACH OF WHICH PUTS THE
 * ACHIEVEMENT OF THE PURPOSE OF THE AGREEMENT AT STAKE, RESPECTIVELY, AND ON THE
 * FULFILMENT OF WHICH THE RECIPIENT THEREFORE MAY RELY ON OR WAS CAUSED BY GROSS
 * NEGLIGENCE OR INTENTIONALLY. ANY FURTHER LIABILITY FOR DAMAGES SHALL -
 * IRRESPECTIVE OF THE LEGAL GROUND - BE EXCLUDED. IN THE EVENT THAT GEMALTO IS
 * LIABLE FOR THE VIOLATION OF A MAJOR CONTRACTUAL OBLIGATION IN THE ABSENCE OF
 * GROSS NEGLIGENCE OR WILFUL CONDUCT, SUCH LIABILITY FOR DAMAGE SHALL BE LIMITED
 * TO AN EXTENT WHICH, AT THE TIME WHEN THE RESPECTIVE AGREEMENT IS CONCLUDED,
 * GEMALTO SHOULD NORMALLY EXPECT TO ARISE DUE TO CIRCUMSTANCES THAT THE PARTIES
 * HAD KNOWLEDGE OF AT SUCH POINT IN TIME. GEMALTO SHALL IN NO EVENT BE LIABLE
 * FOR INDIRECT AND CONSEQUENTIAL DAMAGES OR LOSS OF PROFIT. GEMALTO SHALL IN NO
 * EVENT BE LIABLE FOR AN AMOUNT EXCEEDING Ъ 20,000.00 PER EVENT OF DAMAGE. WITHIN
 * THE BUSINESS RELATIONSHIP THE OVERALL LIABILITY SHALL BE LIMITED TO A TOTAL
 * OF Ъ 100,000.00. CLAIMS FOR DAMAGES SHALL BECOME TIME-BARRED AFTER ONE YEAR AS
 * OF THE BEGINNING OF THE STATUTORY LIMITATION PERIOD. IRRESPECTIVE OF THE
 * LICENSEE'S KNOWLEDGE OR GROSS NEGLIGENT LACK OF KNOWLEDGE OF THE CIRCUMSTANCES
 * GIVING RISE FOR A LIABILITY ANY CLAIMS SHALL BECOME TIME-BARRED AFTER FIVE
 * YEARS AS OF THE LIABILITY AROSE. THE AFOREMENTIONED LIMITATION OR EXCLUSION
 * OF LIABILITY SHALL NOT APPLY IN THE CASE OF CULPABLE INJURY TO LIFE, BODY OR
```

```
 * HEALTH, IN CASE OF INTENTIONAL ACTS, UNDER THE LIABILITY PROVISIONS OF THE
     * GERMAN PRODUCT LIABILITY ACT (PRODUKTHAFTUNGSGESETZ) OR IN CASE OF A
 * CONTRACTUALLY AGREED OBLIGATION TO ASSUME LIABILITY IRRESPECTIVE OF ANY
 * FAULT (GUARANTEE).
 *
  * IN THE EVENT OF A CONFLICT BETWEEN THE PROVISIONS OF THIS AGREEMENT AND
  * ANOTHER AGREEMENT REGARDING THE SOURCE CODE  OF THIS SOFTWARE (IMLET, LIBLET,
 * BATCH FILES, PROJECT FILES) (EXCEPT THE GENERAL TERMS AND CONDITIONS OF
 * GEMALTO) THE OTHER AGREEMENT SHALL PREVAIL.
 *
 */

import gpio.GPIO5V_Factory;
import gpio.InPort5V;
import gpio.OutPort5V;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Vector;

import javax.microedition.io.Connector;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

//import AtCmdDemo.AtCmdDemo.Listener;

import com.cinterion.io.ADC;
import com.cinterion.io.ATCommand;
import com.cinterion.io.ATCommandFailedException;
import com.cinterion.io.I2cBusConnection;
import com.cinterion.io.InPort;

public class Test_routine extends MIDlet {


  public Test_routine() {
    // TODO Auto-generated constructor stub
  }

  protected void destroyApp(boolean unconditional)
      throws MIDletStateChangeException {
    notifyDestroyed();
  }

  protected void pauseApp() {
    // TODO Auto-generated method stub
  }

  protected void startApp() throws MIDletStateChangeException
{
    test();
    destroyApp(true);
  }

  protected void sendSMS()
  {
      try {

          System.out.println("Sending SMS in progress...");

         ATCommand m_Cmd = new ATCommand(false);


String registration_response = m_Cmd.send("AT+CREG?\r"); //
Checking if module is registered to the network

System.out.println(registration_response);

int localy_registered = registration_response.indexOf(",1");
System.out.println(localy_registered);
int roaming_registered = registration_response.indexOf(",5");
if((localy_registered >-1 ) || (roaming_registered >-1)){

  System.out.println("Module registered to the network");

         String Response = m_Cmd.send("AT+cmgf = 1\r");
         System.out.println(Response);
        Response = m_Cmd.send("AT+CMGS=\"+79030189102
\"\r"); //here you put your nummber
         System.out.println(Response);
             Response = m_Cmd.send("(c) Sergey Dronsky" +
(char)26); //here you put text that is inside the SMS
         System.out.println(Response);

}else{
  System.out.println("Module not registered to the network");

}
      } catch (ATCommandFailedException ex) {
         ex.printStackTrace();
      } catch (IllegalStateException ex) {
         ex.printStackTrace();
      } catch (IllegalArgumentException ex) {
         ex.printStackTrace();
      }
  }
```

КОММЕНТАРИЙ СПЕЦИАЛИСТА

Иван Гончаров,
инженер по внедрению холдинга PT Electronics,
ivan.goncharov@ptelectronics.ru

В статье отображены простейшие действия GSM-модема со встроенной Java Me Embedded. Разработчик может по своему усмотрению решить, как использовать модуль более эффективно в своем изделии, например, управлять портами ввода/вывода, принимать и обрабатывать данные с температурных и иных датчиков, с приемопередатчиков ISM-диапазона и Bluetooth-модулей, взаимодействовать напрямую с ГЛОНАСС-приемником. На нашем рынке набирает популярность продукт 3G-модем EHS5, содержащий всю ту же периферию, что и EHS6, а также со встроенной Java. EHS5 полностью совместим с BGS2, что позволяет разработчику разнообразить свое изделие при переходе на 3G без существенных затрат, но с наибольшим функционалом.

```
  /**
   * To run this Test, connect the GPIO6 and GPIO7 pins. Then
the Output
   * should be like this:<br>
   * <b> 0; OutPort: 0; InPort: 0<br>
   * 1; OutPort: 1; InPort: 1<br>
   * 2; OutPort: 0; InPort: 0<br>
   * 3; OutPort: 1; InPort: 1<br>
   * <br>
    * Note:</b> In case of error the InPort values are always 0
or 1.
   */
  private void test() {
    InPort5V inPort = null;
    OutPort5V outPort = null;
     GPIO5V_Factory factory = new GPIO5V_Factory(GPIO5V_
Factory.CONCEPT);

    try {

      int counter = 0;
      Vector outPins = new Vector(1);
      Vector values = new Vector(1);
      outPins.addElement("GPIO6");
      values.addElement(Integer.valueOf("0"));
      outPort = factory.getOutPort5V(outPins, values);

      ADC adc = new ADC(0,0);

         I2cBusConnection cc = (I2cBusConnection) Connector.
open("i2c:0;baudrate=400");

      //int baudrate = cc.getBaudRate();
      InputStream inStream  = cc.openInputStream();
      OutputStream outStream = cc.openOutputStream();

      String data = "<aD21300>";

      outStream.write(data.getBytes(), 0, data.length());
      outStream.flush();

      byte[] inBuf = new byte[12];
      if (inStream.available()>0)
      inStream.read(inBuf);

      inStream.close();
      outStream.close();
      cc.close();

       //Write Transfer Frame, where a = message ID, AE = Slave
Address and write request:  String data = "<aAE000102030405
060708090A0B0C0E0F>";
     // outStream.write(data.getBytes(), 0, data.length());
     // outStream.flush();

      Vector inPins = new Vector(1);

      inPins.addElement("GPIO7");
      //inPins.addElement("GPIO9");

      inPort = factory.getInPort5V(inPins);

      Vector pins1 = new Vector(2);
      pins1.addElement("GPIO11");
      pins1.addElement("GPIO12");

     InPort inport1 = new InPort(pins1);

      System.out.println("GO");

      while (inport1.getValue()==0 ) { //or counter < 40
        System.out.print(counter + "; OutPort: " + counter % 2);
        try {
          outPort.setValue(counter);
        } catch (IOException e1) {
          e1.printStackTrace();
        }
        try {
          System.out.print("; InPort: " + inPort.getValue());
         System.out.print("; ADC: " + adc.getValue()*4.17 + " mV");
          System.out.println("; InPort1: " + inport1.getValue());

        } catch (IOException e) {
          e.printStackTrace();
        }
        try {
          Thread.sleep(20);
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
        counter = 1 + counter;
      }
      outPort.release();
      inPort.release();
      sendSMS();
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Заключение

В статье рассмотрен модуль EHS6 и макетная плата Cinterion Concept Board. Приведены простейшие способы общения с аппаратурой, вывод информации на разъем, считывание данных с разъема и аналого-цифрового преобразователя, посылка SMS из мидлета, установка и использование системы Eclipse для создания и отладки программ на Java ME 3.2. Все это возможно реализовать на базе модуля EHS5, который является продолжением линейки Industrial и полностью совместим с модемами 2G BGS2 и BGS5. Отметим также, что полноценное использование модуля возможно только после детального изучения всей сопутствующей документации – и той, что размещена на сайте, и той, что устанавливается на компьютер.